TUBITAK-UZAY Compulsory Summer Internship Project

DESIGN DOCUMENT

TUBITAK-UZAY-MultiCommSim

2025 SUMMER

EDİZ ARKIN KOBAK – Computer Engineer

TABLE OF CONTENTS

1. Introduction

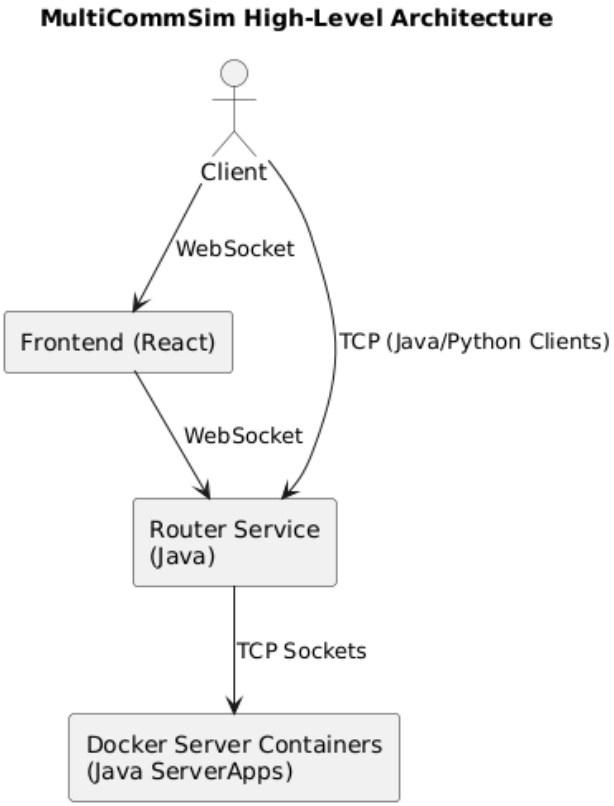
MultiCommSim, TÜBİTAK UZAY'daki uzay simülasyon cihazında geliştirilen, tek bir IP ve port üzerinden çoklu client-server iletişimini sağlayan bir sistemdir.
Server uygulamaları Docker konteynerlerinde izole şekilde çalışır. Router servisi, tüm client bağlantılarını tek port üzerinden yönetir ve ilgili server konteynerlarına iletir.
Projede Java kullanılırken, test ve otomasyonlar Python ile gerçekleştirilir. İleride kullanıcıların etkileşimi için React tabanlı WebSocket destekli bir basit frontend arayüzü önerilmektedir.

2. Software Architecture

2.1 Architectural Layers

Katman	Teknoloji/ Dil	Sorumluluklar
Application Layer	*Java (Client, Server)*	Client ve Server uygulamaları, TCP ve WebSocket üzerinden iletişim kurar.
Router Layer	*Java*	Tek port üzerinde tüm client mesajlarını kabul eder, oturumlara ayırır, Docker konteynerlarına yönlendirir. WebSocket endpoint barındırabilir.
Container Layer	*Docker*	Her server örneği izole bir konteyner içinde çalışır.
Frontend Layer	*React+WebSocket*	(Opsiyonel) Tek ekranlı, gerçek zamanlı mesajlaşma ve oturum izleme arayüzü.
Testing Layer	*Python*	Socket tabanlı senaryo testleri ve otomasyon.
Deployment Layer	*Docker Compose / Bash*	Ortam kurulumu ve servis yönetimi.

2.2 High-level Architecture Diagram



2.3 Communication Strategy: React WebSocket ve TCP Socket İlişkisi

Durum

- Server tarafı uygulamalar Java TCP socket tabanlı iletişim kullanmaktadır.
- Web tarayıcıları (React gibi frontend teknolojileri) doğrudan TCP socket bağlantısı açamazlar; bu, tarayıcı güvenlik politikaları nedeniyle mümkün değildir.

Sonuç

- React frontend, TCP socket ile doğrudan haberleşemez.

Çözüm Önerileri

1. *RouterService'de WebSocket Endpoint Açılması:*
    - RouterService, mevcut TCP socket server fonksiyonelliğine ek olarak bir WebSocket endpoint barındırır.
    - React frontend bu WebSocket endpoint'e bağlanır.
    - RouterService gelen WebSocket mesajlarını uygun TCP socket bağlantılarına çevirir ve tersi yönlü iletişimi sağlar.
    - Böylece React arayüzü ve TCP tabanlı backend arasında WebSocket ↔ TCP proxy görevi görür.
2. *TCP Client Uygulamaları:*
    - TCP tabanlı iletişim için ayrı Java veya Python client uygulamaları geliştirilir.
    - Bu uygulamalar RouterService'in TCP portuna bağlanarak direkt iletişim kurar.
3. *React UI'nın Rolü:*
    - React uygulaması esas iş mantığına müdahale etmeden, izleme, test senaryoları başlatma ve gerçek zamanlı mesaj görüntüleme amacıyla WebSocket üzerinden RouterService'e bağlanır.

ÖZET

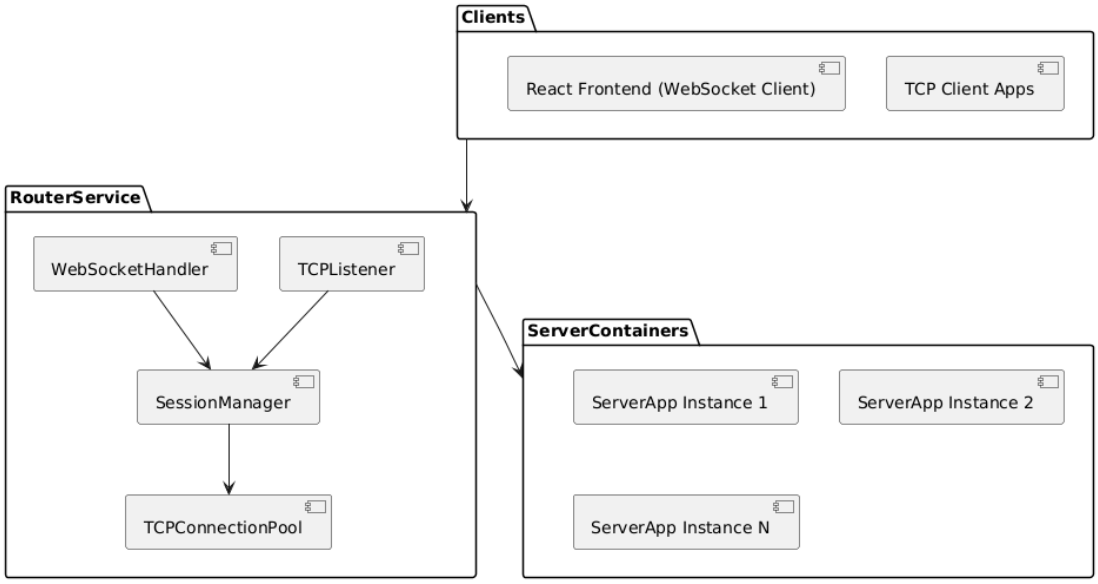
Katman	İletişim Protokolü	Kullanım Amacı
React Frontend	WebSocket	UI, izleme, test, gerçek zamanlı etkileşim
RouterService	WebSocket ↔ TCP Proxy	Mesajları WebSocket ve TCP arasında dönüştürür
Server Containers	TCP Socket	Uygulama iş mantığı, container içinde çalışır
TCP Client Apps	TCP Socket	Alternatif client uygulamalar için

3. Structural Design

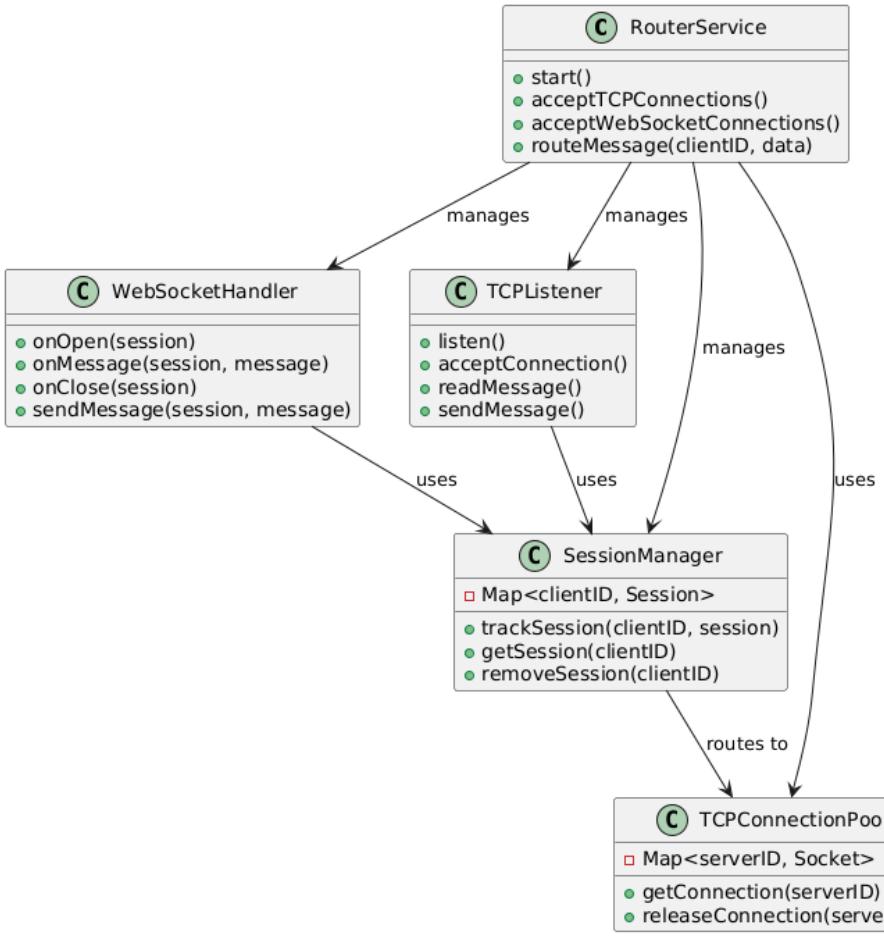
3.1 Module Breakdown

Module	Language	Description
RouterService	Java	Gelen tüm client bağlantılarını tek port üzerinden dinleyen ana servis. Hem TCP hem WebSocket bağlantılarını kabul eder, oturumları yönetir ve mesajları ilgili server container'lara yönlendirir.
WebSocketHandler	Java	RouterService içinde WebSocket bağlantılarını yöneten alt modül. React gibi WebSocket client'ların bağlanmasını sağlar ve mesajları TCP socket'lere iletir.
TCPListener	Java	RouterService içindeki TCP socket listener. Java/Python TCP client uygulamalarından gelen bağlantıları kabul eder ve mesajları işler.
SessionManager	Java	Client oturumlarının takibini yapar. ClientID bazında bağlantıları yönetir, WebSocket ve TCP oturumları arasında ilişki kurar.
TCPConnectionPool	Java	Docker konteyner içindeki ServerApp'lere ait TCP socket bağlantılarını yönetir. Mesajların ilgili server container'a iletilmesini sağlar.
ServerApp	Java	Docker container içinde çalışan server uygulaması. RouterService'den gelen mesajları işler ve yanıt üretir.
ClientApp	Java	TCP socket üzerinden RouterService'e bağlanıp mesaj gönderen ve yanıt alan örnek client uygulaması.
FrontendUI	React (JavaScript/TypeScript)	WebSocket üzerinden RouterService'e bağlanan, tek ekranlı, basit ve kullanıcı dostu arayüz. ClientID girişi, mesaj gönderme, gelen cevapların görüntülenmesi gibi işlevler sağlar.
TestClient.py	Python	Otomatik test senaryolarını çalıştıran, TCP veya WebSocket tabanlı client simülatörü. Farklı senaryoları test eder, performans ve yük testi yapabilir.
compose.yaml	YAML	Docker Compose yapılandırması. RouterService, ServerApp containerları, varsa FrontendUI container'ı ve network ayarlarını tanımlar.
Deployment Scripts	Bash / Shell	Ortam kurulumu, container başlatma, durdurma ve bakım işlemleri için scriptler.

3.2 Component Diagram



3.3 Class Diagram



4. Behavioral Design

4.1 Use Case: Client Sends Message

1. Client uygulaması, RouterService'in WebSocket veya TCP portuna bağlanır (örneğin port 6003).

- React frontend, WebSocket protokolü üzerinden bağlanır.
- Java/Python TCP client ise doğrudan TCP socket ile bağlanır.

2. RouterService, gelen bağlantıyı clientID ile tanımlar ve oturumu başlatır.

- Gelen mesaj içindeki clientId alanı kullanılarak oturum yönetimi yapılır.

3. Client mesajını gönderir (örnek JSON): { "clientId": "client_42", "message": "Hello" }

4. RouterService, clientID'ye göre ilgili Server container'ın TCP socket bağlantısına mesajı iletir.

- WebSocket mesajları RouterService içinde TCP socket'e dönüştürülür.
- TCPListener aracılığıyla gelen TCP mesajları da aynı şekilde yönlendirilir.

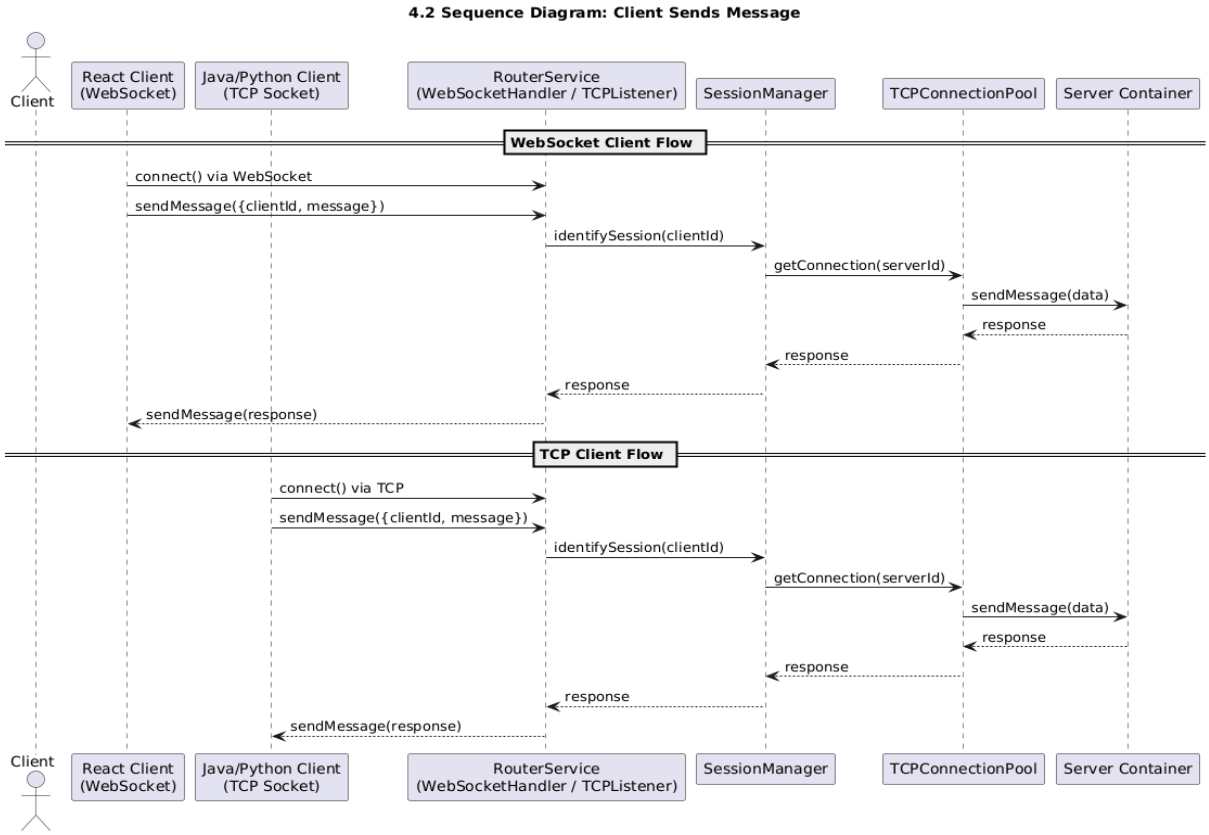
5. Server container, mesajı işler ve yanıt üretir.

6. Yanıt, RouterService üzerinden client'a geri döner.

- WebSocket client'a WebSocket mesajı olarak, TCP client'a TCP socket mesajı olarak iletilir.

7. (Opsiyonel) Frontend arayüzde ilgili client oturumu için gönderilen mesaj ve alınan yanıt anlık olarak görüntülenir.

4.2 Sequence Diagram



5. Data Flow & Integration

5.1 Input Sources

- Client uygulamaları (Java/TCP veya React/WebSocket)
- TCP/WebSocket üzerinden gelen JSON mesajlar

5.2 Internal Flow

- RouterService tek portu dinler.
- Mesaj header'ından clientID çıkarılır.
- Mesaj ilgili server container socket'a yönlendirilir.
- Server container'dan gelen yanıt Router üzerinden client'a iletilir.
- Frontend WebSocket client mesaj ve yanıtları alır, gerçek zamanlı gösterir.
- Gelen WebSocket mesajları WebSocketHandler tarafından alınıp, clientID parse edilir.
- SessionManager ile clientID'ye karşılık gelen TCP socket bulunur.
- Mesaj TCP socket'a iletilir, yanıt alınır, WebSocket client'a gönderilir.
- TCP Client mesajları doğrudan TCPListener tarafından alınır, aynı mekanizma izlenir.

---

6. Graphical User Interface (Optional)

- Tek Ekranlı React UI
    - Client ID giriş alanı
    - Mesaj gönderme kutusu
    - Gönderilen ve alınan mesajların listesi (log)
    - Bağlantı durumu göstergesi
    - Basit stil, minimum fonksiyonellik
- Mesajlaşma ve oturum yönetimi tek ekran üzerinden yapılır.
- React tarafı WebSocket üzerinden RouterService'e bağlanır.
- RouterService Java tarafında WebSocket endpoint implementasyonu gerekir.

---

7. Deployment Notes

- Server uygulamaları Docker container olarak çalışır.
- RouterService, local makinede ya da ayrı bir container'da, port 6003'e bind olur.
- Docker Compose veya bash script ile tüm containerlar ayağa kaldırılır.
- Client uygulamalar RouterService'in 6003 numaralı portuna bağlanır.
- Testler Python socket scriptleri ile gerçekleştirilir.
- Frontend UI istenirse React ile geliştirilip ayrı bir container'da veya statik servis olarak sunulabilir.

---

8. Future Work

- Client kimlik doğrulama ve yetkilendirme (auth token)
- Kubernetes ile container orchestration
- Veri iletimi için Protobuf/GRPC desteği
- WebSocket ile gerçek zamanlı yüksek performanslı iletişim iyileştirmeleri
- UI üzerinde gelişmiş oturum yönetimi ve log filtreleme
- Performans izleme ve logging entegrasyonları
- WebSocket performans iyileştirmeleri
- Auth token entegrasyonu WebSocket ve TCP için
- Kubernetes ile otomatik scaling

Prepared by: Ediz Arkın Kobak
Date: 02.07.2025